JDDPS Stubs, Data, and Activities

Eric H. Neilsen, Jr.

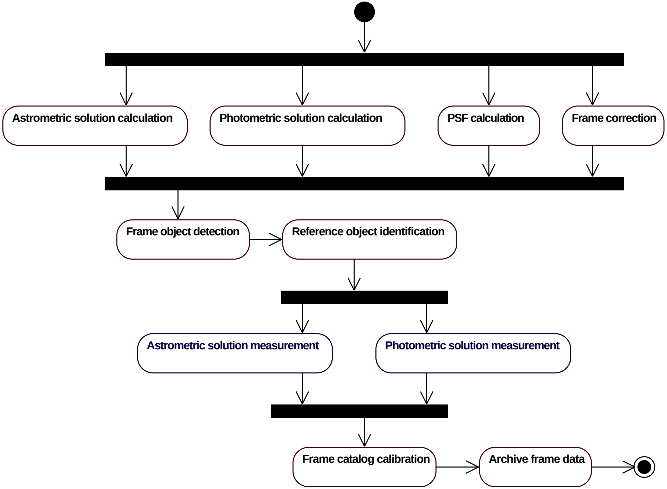
FNAL

29 June 2010

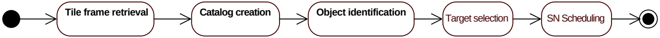
Outline

- Activity diagrams
- Class diagrams
- File descriptions
- Description of the universal stub
- Database schema
- The state of the data model

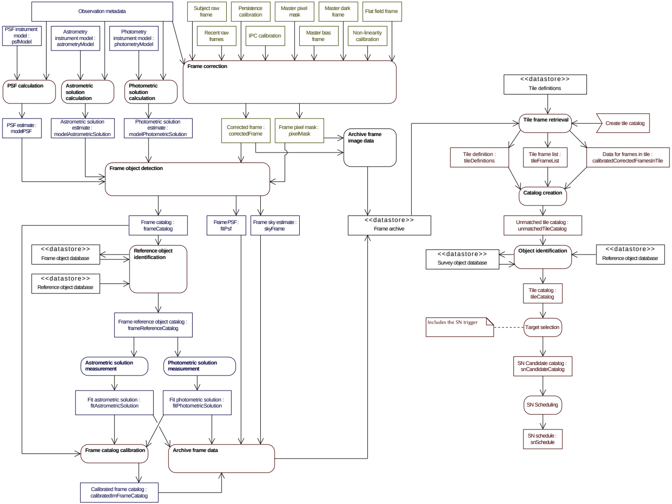
Frame processing


Astrometric solution calculation
Photometric solution calculation
PSF calculation
Frame correction
Frame object detection
Reference object identification
Astrometric solution measurement
Photometric solution measurement
Frame catalog calibration
Archive frame data

Tile processing



Object flow



Observation metadata

Subject raw frame

Persistence calibration

Master pixel mask

Master dark frame

Flat field frame

PSF instrument model : psfModel

Astrometry instrument model : astrometryModel

Photometry instrument model : photometryModel

Recent raw frames

IPC calibration

Master bias frame

Non-linearity calibration

Frame correction

PSF calculation

Astrometric solution calculation

Photometric solution calculation

PSF estimate : modelPSF

Astrometric solution estimate : modelAstrometricSolution

Photometric solution estimate : modelPhotometricSolution

Corrected frame : correctedFrame

Frame pixel mask : pixelMask

Archive frame image data

Frame object detection

Frame catalog : frameCatalog

Frame PSF : fitPsf

Frame sky estimate : skyFrame

<<datastore>>
Frame archive

<<datastore>>
Frame object database

<<datastore>>
Reference object database

Reference object identification

Frame reference object catalog : frameReferenceCatalog

Astrometric solution measurement

Photometric solution measurement

Fit astrometric solution : fitAstrometricSolution

Fit photometric solution : fitPhotometricSolution

Frame catalog calibration

Archive frame data

Calibrated frame catalog : calibratedImFrameCatalog

<<datastore>>
Tile definitions

Tile frame retrieval

Create tile catalog

Tile definition : tileDefinitions

Tile frame list : tileFrameList

Data for frames in tile : calibratedCorrectedFramesInTile

Catalog creation

Unmatched tile catalog : unmatchedTileCatalog

<<datastore>>
Survey object database

Object identification

<<datastore>>
Reference object database

Tile catalog : tileCatalog

Includes the SN trigger

Target selection

SN Candidate catalog : snCandidateCatalog

SN Scheduling

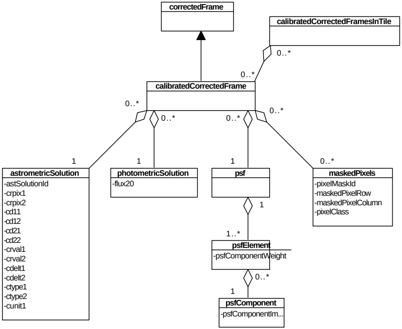
SN schedule : snSchedule

The use of class diagrams

This data model uses class diagrams as a purely abstract tool to organize related atomic entries in the data dictionary. Programs that make use of the data model are not expected to implement code according to these classes.

The “classes” are used instead as a shorthand used when defining file formats and database table schema.

Example diagram: the calibrated corrected frame



File descriptions

There are three potential types of data in FITS files: headers, images, and tables. Each header keyword, array of image data, and table column corresponds to a member of class. Usually, when one member of a class is present as a table column or header keyword, all members of that class are.

Mapping classes to FITS files

- The classes in the class diagrams group related atomic data. If one member of a class appears in a FITS header or table, the others usually do as well. When describing FITS files, we can use a class name as a shorthand for a set of values.
- If classes and file templates are specified in form that can be read and interpreted by a script, simple scripts can be used to verify, create, and document FITS files.

Representing class definitions in yaml

```
field: &field
  fieldId: &fieldId
    name: fieldId
    keyword: FIELDID
    type: 1J
    format: int
    comment: unique field id
  fieldRa: &fieldRa
    name: fieldRa
    keyword: FIELDRA
    format: 1D
    type: double
    units: degrees
    sampleValue: 12.0
    comment: the field RA (J2000) in decimal degrees
  fieldDec: &fieldDec
    name: fieldDec
    keyword: FIELDRA
    format: 1D
    type: double
    units: degrees
    sampleValue: 12.0
    comment: the field RA (J2000) in decimal degrees
  fieldPa: &fieldPa
    name: fieldPA
```

Representing file specifications in yaml

Files can now be specified in terms of the defined classes:

```
tileDetections:
  nameTemplate: tcat-%d-%d.fits
  nameParams:
    - *tileId
    - *imagingProcId
  header:
    <<: *tileDefinition
    imagingProcId: *imagingProcId
  HDUs:
    - name: tileDetections
      header:
        <<: *tileDefinition
        imagingProcId: *imagingProcId
      table:
        <<: *imCatalogEntry
        <<: *referenceCatalogEntry
        <<: *refObj
    - *periodTileDetections
```

Documenting FITS file specifications

A short python script that parses yaml can now generate documentation for FITS files:

Sample FITS file specification, p1

1 tileDetections

1.1 File name

Template

tcat-%d-%d.fits

Template parameters

tileId A unique identifier for the tile

imagingProcId a unique ID for an execution of an imaging actor

1.2 HDU 1

Header

Name	Keyword	Type	Description
crval2	CRVAL2	float	See the WCS defn in the FITS defn, v3
crpix1	CRPIX1	float	See the WCS defn in the FITS defn, v3
crpix2	CRPIX2	float	See the WCS defn in the FITS defn, v3
cd22	CD2$_2$	float	See the WCS defn in the FITS defn, v3
cd21	CD2$_1$	float	See the WCS defn in the FITS defn, v3
frameId	FRAMEID	int	the unique ID of the corrected frame
cunit1	CUNIT1	string	See the WCS defn in the FITS defn, v3
ctype2	CTYPE2	string	See the WCS defn in the FITS defn, v3
ctype1	CTYPE1	string	See the WCS defn in the FITS defn, v3
cunit2	CUNIT2	string	See the WCS defn in the FITS defn, v3
photSolutionId	PHTSOLN	int	unique index for a photometric solution
tileRows	TILEROWS	int	The number of rows in a tile
tileColumns	TILECOLS	int	The number of columns in a tile
imagingProcId	IMPROCID	int	a unique ID for an execution of an imaging actor
cdelt1	CDELT1	float	See the WCS defn in the FITS defn, v3
cdelt2	CDELT2	float	See the WCS defn in the FITS defn, v3
astSolutionId	ASTSLNID	int	a unique id for an astrometric solution
flux20	FLUX20	float	the number of DN for a magnitude 20 object
cd12	CD1$_2$	float	See the WCS defn in the FITS defn, v3
cd11	CD1$_1$	float	See the WCS defn in the FITS defn, v3
crval1	CRVAL1	float	See the WCS defn in the FITS defn, v3
tileId	TILEID	int	A unique identifier for the tile

Sample FITS file specification, p2

Table

Column name	Format	Units	Description
origin	20A		the name of the survey the detected the object
magErr	1E		the 1 std uncertainty in the magnitude
phi	1E		the position angle of the major axis (+N through E) in degrees
refDecErr	1D	degrees	the 1 std uncertainty in object dec in decimal degrees
raErr	1D	degrees	the 1-sigma uncertainty in object RA in decimal degrees
refVariability	1E		the estimated variability (magnitudes)
frameId	1J		the unique ID of the corrected frame
refMagErr	1E		the 1 std uncertainty in the magnitude
radius	1E		the scale radius in arcseconds
ellipticity	1E		the ellipticity
mag	1E		the magnitude
objId	1J		the databases unique ID for this detection of this object
refDec	1D	degrees	the object DEC (J2000) in decimal degrees
imagingProcId	1J		a unique ID for an execution of an imaging actor
imDetectionId	1J		a unique ID for a detection
refRa	1D	degrees	the object RA (J2000) in decimal degrees
decErr	1D	degrees	the 1 std uncertainty in object dec in decimal degrees
refObjId	1J		the unique ID for a reference object
astSolutionId	1J		a unique id for an astrometric solution
ellipticityErr	1E		the uncertainty in the ellipticity
refMag	1E		the magnitude
ra	1D	degrees	the object RA (J2000) in decimal degrees
refRaErr	1D	degrees	the 1-sigma uncertainty in object RA in decimal degrees
phiErr	1E		the uncertainty in the position angle
radiusErr	1E		the 1 std uncertainty in the scale radius (asec)
dec	1D	degrees	the object DEC (J2000) in decimal degrees
photSolutionId	1J		unique index for a photometric solution

Constructing actor stubs

- The stub for each actor must read one or more files, and write one or more files.
- The format details for these files are determined by the data model.
- A single program capable of reading the data model should therefore be able to act as a stub for any actor.

Actor configuration

In the SDSS pipeline, configuration took the form of two files:

- a plan file, which held configuration data likely to be different for each execution (the names of the input data files, for example), and
- a “param” file, which held other, more stable tunable parameters.

SDSS used Yanny par files, which are convenient but obscure.
I use yaml files.

Sample plan and param files

plan

```
parameters: findObjectsParam-v12.yml
inputDataDir: /data2/foo/bar
frameId: 102
mjd: 56921
```

param

```
detThresh: 5 #detection threshhold in noise sigma
minArea: 3 #the minimum pixels an object must cover
```

Execution

```
bash$ findObjects findObjectsPlan-102.yml
```

Supplementing the param for a universal actor

PLAN

```
parameters: findObjectsParam-v12.yml
inputDataDir: /data2/foo/bar
frameId: 102
mjd: 56921
```

PARAM

```
dataModel: /foo/bar/dataModel.yml
detThresh: 5 #detection threshhold in noise sigma
minArea: 3 #the minimum pixels an object must cover

inputFiles:
  - fitsType: correctedFrame

outputFiles:
  - fitsType: imFrameCatalog
  - fitsType: skyFrame
```

EXECUTION

```
bash$ universalStub correctFramePlan.yml
```

Approaches to creating a database schema

Map Classes Each class in the class diagram can be mapped to a table in the database schema; members correspond to columns, and rows to objects of that class. In other words, follow typical ORM practices. (Note that PostgreSQL supports table inheritance.)

Map Tables Each table or header in a FITS file can be mapped to a table in the database schema.

Hand Generated Database table schema can be specified following the approach used to specify FITS tables.

The database will be easier to use if each table or header in a FITS file corresponds to either a true table or a view in the database. The “Map Tables” approach provides this by definition.

Created items

- Most class diagrams (47)
- A script that maps class diagrams to FITS files
- Most FITS file specifications (24 from the NIR object flow diagram, 40 including slitless spectroscopy)
- A universal stub script
- A database schema that directly maps the class definitions

Needed items

- Classes and file specifications for inputs to frame correction
- Modeling of the prototype slitless spectroscopy actors
- A script to synchronize class diagrams from diagramming tool with the yaml file
- Integration with the slitless spectroscopy prototype
- A document that introduces and presents the diagrams and specifications
- A FITS file format verification script
- Additional pairs of eyes, and significant refinement.